The Tiny Language

Compiler Project

Team members :

- **Samaa sabry ab del Wahab mohamed 20201700362**
- **Salwa ahmed sayed ahmed osman 20201700356**

Task 1:

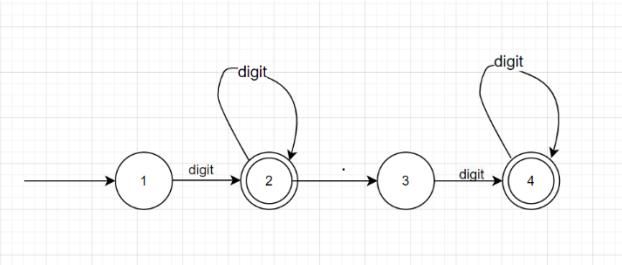
Regular expression rules of tiny languages

1. Number: any sequence of digits and maybe floats (e.g. 123 | 554 | 205 | 0.23 | …)

   digit = [0-9]

   nat = digit+(/. Digit+)?

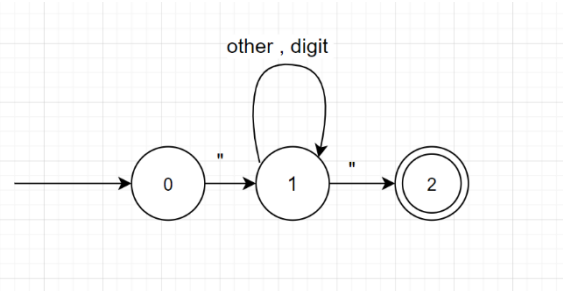
A DFA of NUMBER:



2. String: starts with double quotes followed by any combination of characters and digits then ends with double quotes (e.g. “Hello” | “2nd + 3rd” | …)

   RE: = “(. | digit)*”

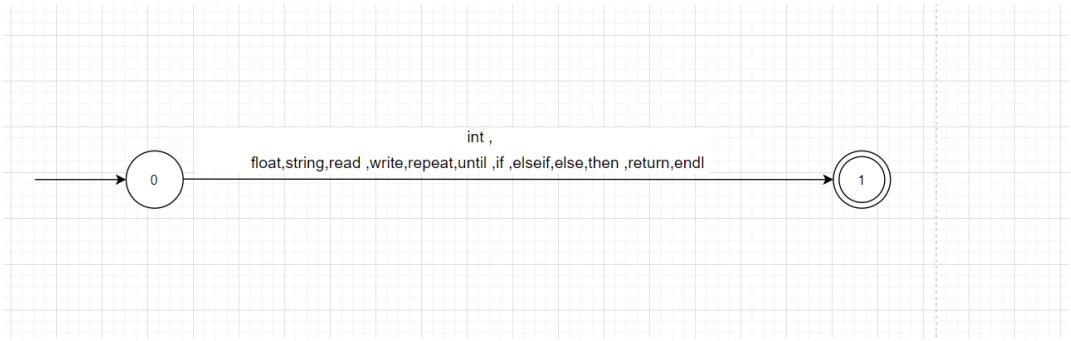
   A DFA of STRING:



3. Reserved Keywords: int | float | string | read | write | repeat | until | if | elseif | else | then | return | endl

RE: =(int | float | string | read | write | repeat | until | if | elseif | else | then | return | endl)

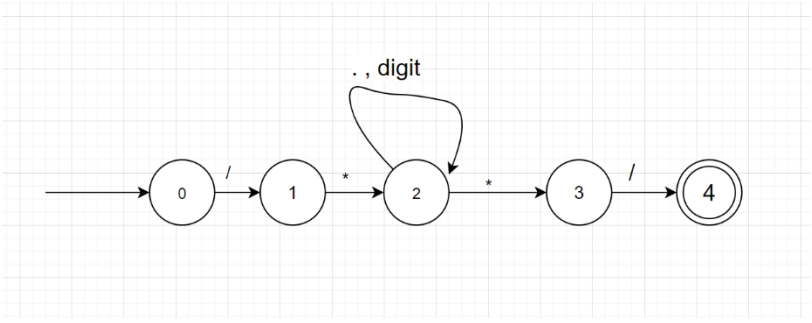
A DFA of Reserved word:



4. Comment_Statement: starts with /* followed by any combination of characters and digits then ends with */ (e.g. /*this is a comment*/ | …)

RE: =/(*) (. | digit)* (*)/

A DFA of comment statement:

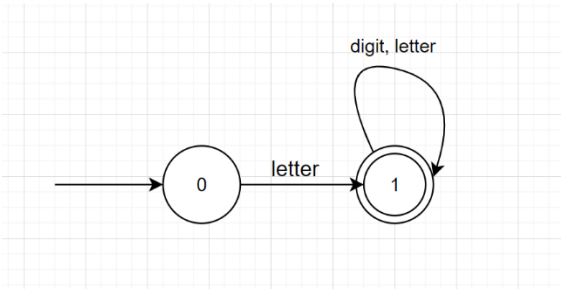


5. Identifiers: starts with letter then any combination of letters and digits. (e.g. x | val | counter1 | str1 | s2 | …)

Letter RE: = [_ | a-z |A-Z]
Digit RE: = [0-9]

RE: =Letter (Letter | digit) *

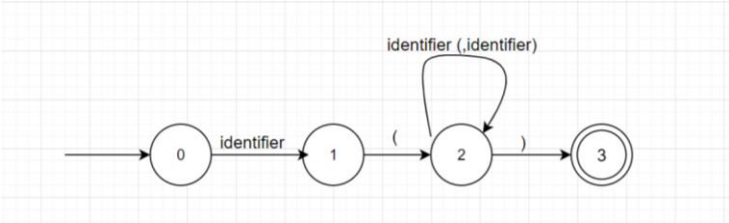
A DFA of Identifier:



6. Function_Call: starts with Identifier then left bracket “(“ followed by zero or more Identifier separated by “,” and ends with right bracket “)”. (e.g. sum(a,b) | factorial(c) | rand() | …)

RE: =identifier “(“ (identifier(, identifier)*)* “)”

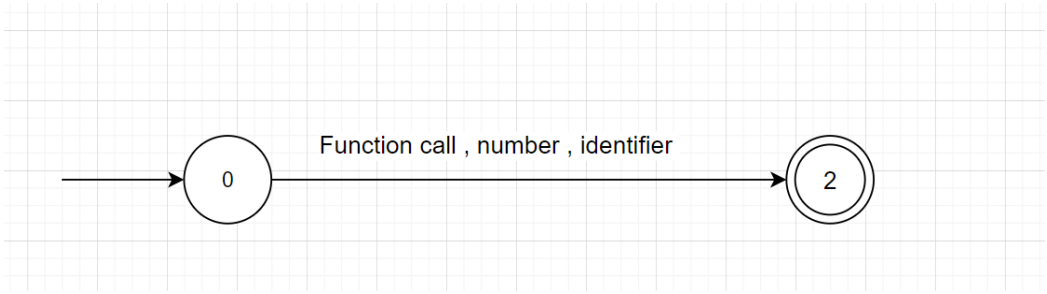
A DFA of function call:



7. Term: maybe Number or Identifier or function call. (e.g. 441 | var1 | sum(a,b) | …)

   RE: =number | identifier | function call

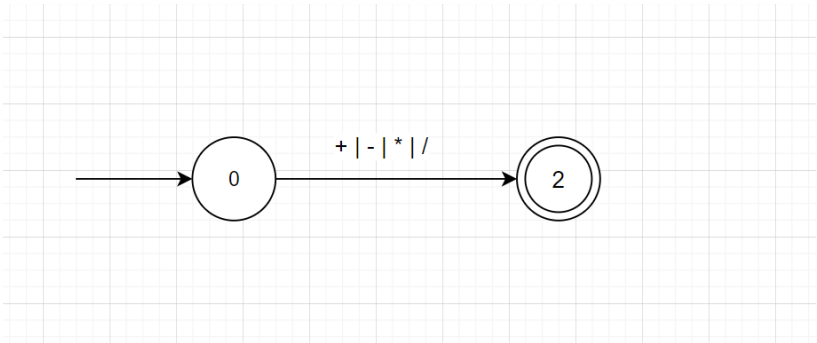
   A DFA of Term:



8. Arithmatic_Operator: any arithmetic operation (+ | - | * | /)

   RE: =+ | - | * | /

   A DFA of Arithmatic_Operator**:**

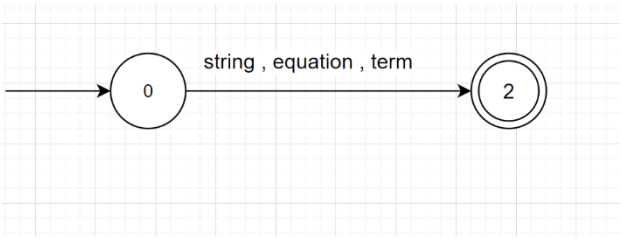


9. Equation: starts with Term or left bracket "(" followed by one or more Arithmatic_Operator and Term. with right bracket ")" for each left bracket (e.g. 3+5 | x +1 | (2+3)*10 | …)

10. Expression: may be a String, Term or Equation (e.g. “hi” | counter | 404 | 2+3 | …)

RE: = string | term | equation

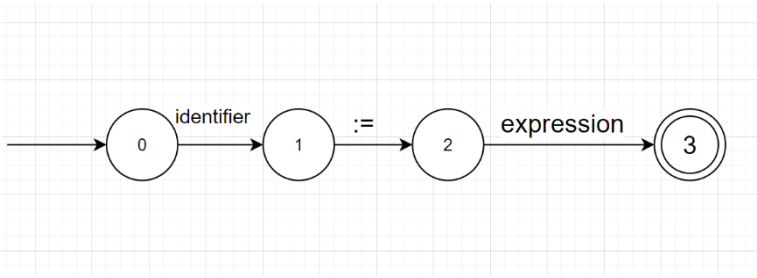
A DFA of expression **:**



11. Assignment_Statement: starts with Identifier then assignment operator “:=” followed by Expression (e.g. x := 1 | y:= 2+3 | z := 2+3*2+(2-3)/1 | …)

RE: = Identifier (:=)expression

A DFA of expression **:**



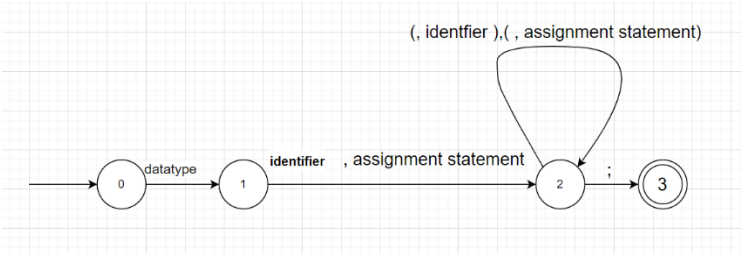
12. Datatype: set of reserved keywords (int, float, string)

RE: = int | float | string

13. Declaration_Statement: starts with Datatype then one or more identifiers (assignment statement might exist) separated by coma and ends with semi-colon. (e.g. int x; | float x1,x2:=1,xy:=3; | …)

RE: = **datatype (identifier|** assignment statement) (, **(identifier|** assignment statement))* ;

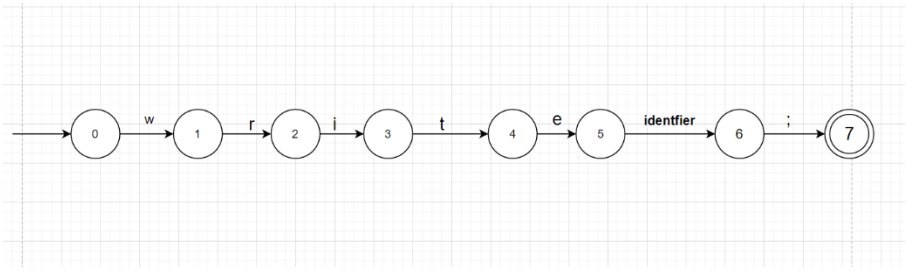
A DFA of Declaration_Statement:



14. Write_Statement: starts with reserved keyword “write” followed by an Expression or endl and ends with semi-colon (e.g. write x; | write 5; | write 3+5; | write “Hello World”; | …)

RE: = **write (expression | endl);**

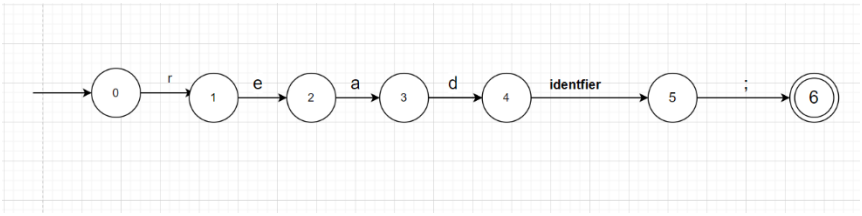
A DFA of Write_Statement:



15. Read_Statement: starts with reserved keyword “read” followed by an Identifier and ends with semi-colon (e.g. read x; | …)

RE: = **read** (Identifier) ;

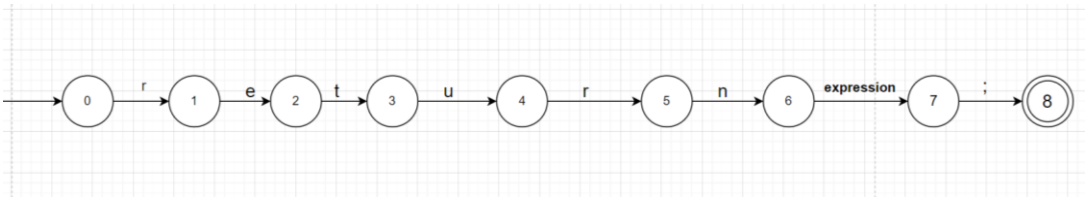
A DFA of Read_Statement:



16. Return_Statement: starts with reserved keyword “return” followed by Expression then ends with semi-colon (e.g. return a+b; | return 5; | return “Hi”; | …)

RE: = **return** (expression) ;

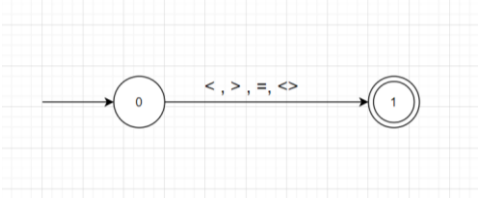
A DFA of return_Statement:



17. Condition_Operator: (less than “<” | greater than “>” | is equal “=” | not equal “<>”)

RE: = **<| > | = |<>**

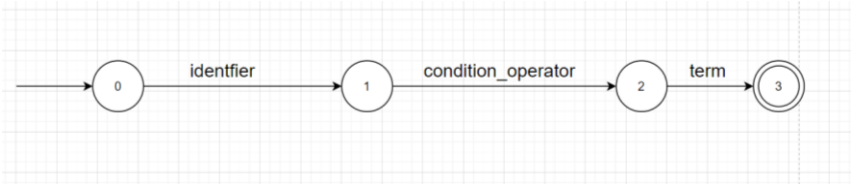
A DFA of condition_operator:



18. Condition: starts with Identifier then Condition_Operator then Term (e.g. z1 <> 10)

RE: = **identifier condition_operator term**

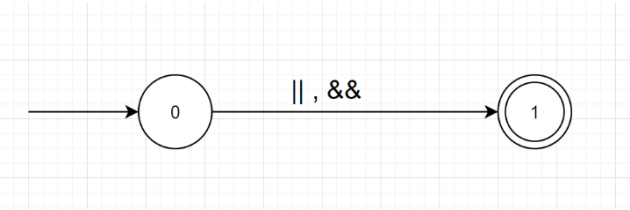
A DFA of condition:



19. Boolean_Operator: AND operator “&&” and OR operator “||”

RE: = (&& | ||)

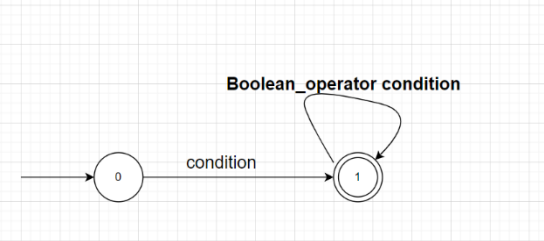
A DFA of Boolean_Operator:



20. Condition_Statement: starts with Condition followed by zero or more Boolean_Operator and Condition (e.g. x < 5 && x > 1)

RE: = condition (Boolean_operator condition)*

A DFA of condition_statement:



21. If_Statement: starts with reserved keyword “if” followed by Condition_Statement then reserved keyword “then” followed by set of Statements (i.e. any type of statement: write, read, assignment, declaration, …) then Else_If_Statment or Else_Statment or reserved keyword “end”

22. Else_If_Statement: same as if statement but starts with reserved keyword “elseif”

RE: = condition (Boolean_operator condition)*

23. Else_Statement: starts with reserved keyword “else” followed by a set of Statements then ends with reserved keyword “end”

24. Repeat_Statement: starts with reserved keyword “repeat” followed by a set of Statements then reserved keyword “until” followed by Condition_Statement

RE statements: = Assignment_Statement |Declaration_Statement | Write_Statement | Read_Statement | Return_Statement | Condition_Statement

RE Repeat_Statement**:** = **repeat statements until condition_statement**

25. FunctionName: same as Identifier

26. Parameter: starts with Datatype followed by Identifier (e.g. int x)

RE: = **datatype identifier**

27. Function_Declaration: starts with Datatype followed by FunctionName followed by "(" then zero or more Parameter separated by "," then ")" (e.g. int sum(int a, int b) | …)

RE: = **datatype identifier "(" (parameter(, parameter)*)* ")"**

28. Function_Body: starts with curly bracket "{" then a set of Statements followed by Return_Statement and ends with "}"

RE: = **{ statements return_statement }**

29. Function_Statement: starts with Function_Declaration followed by Function_Body

RE: = **function_declration function_body**

30. Main_Function: starts with Datatype followed by reserved keyword "main" then "()" followed by Function_Body

RE: = **datatype main () function_body**

31. Program: has zero or more Function_Statement followed by Main_Function

RE: = (function_statements)* main_function